DOCUMENT RESUME

ED 458 245 TM 033 406

AUTHOR Luecht, Richard M.
TITLE Capturing, Codifying and Scoring Complex Data for Innovative, Computer-Based Items.
PUB DATE 2001-04-00
NOTE 30p.; Paper presented at the Annual Meeting of the National Council on Measurement in Education (Seattle, WA, April 11-13, 2001).
PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Certification; *Coding; *Computer Assisted Testing; Data Collection; Scaling; *Scoring; *Test Items

ABSTRACT

The Microsoft Certification Program (MCP) includes many new computer-based item types, based on complex cases involving the Windows 2000 (registered) operating system. This Innovative Item Technology (IIT) has presented challenges beyond traditional psychometric considerations such as capturing and storing the relevant response data from examinees, codifying it, and scoring it. This paper presents an integrated, empirically based data-systems approach to processing complex scoring rules and examinee response data for IIT cases and items, highlighting data management considerations, use of various psychometric analyses to improve the coding and scoring rules, and some of the challenges of scaling the data, using a partial credit item response theory model. Empirical examples from the Microsoft Innovative Item Types project are used to illustrate practical problems and solutions. (Author/SLD)

Reproductions supplied by EDRS are the best that can be made
from the original document.

ED 458 245

PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL HAS BEEN GRANTED BY

R. Luecht

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

1

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Capturing, Codifying and Scoring Complex Data for Innovative, Computer-based Items[1]

Richard M. Luecht
University of North Carolina at Greensboro

April 2001

TM033406

BEST COPY AVAILABLE

[1] Paper presented at the Annual Meeting of the National Council on Measurement in Education, Seattle, WA, April 2001. Symposium: *Implementing Innovative Measurement Technology in a Computer-Based Certification Testing Program.*

Abstract

The Microsoft Certification Program (MCP) includes many new computer-based item types, based on complex cases involving the Windows 2000® operating system. This Innovative Item Technology (IIT) has presented challenges that go beyond traditional psychometric considerations—i.e., capturing and storing the relevant response data from examinees, codifying it, and scoring it. This paper presents an integrated, empirically-based, data systems approach to processing complex scoring rules and examinee response data for IIT cases and items, highlighting data management considerations, use of various psychometric analyses to improve the coding and scoring rules, and some of the challenges of scaling the data, using a partial credit IRT model. Empirical examples from the Microsoft Innovative Item Types project are used to illustrate practical problems and solutions.

Introduction

The Microsoft Certification Program (MCP) includes many new computer-based item types, implemented using testing software components termed Innovative Item Technology (IIT). These new item types are named aptly by their interactive response capabilities—e.g., "create-a-tree", "drag-and-connect", and "build-list-and-reorder" (Fitzgerald, 2001). A more mundane, academic description would be that these IIT items require responses that involve having the examinee select multiple entities, compound sets of entities, and, in some cases, specify relations among those entities or sets. In any case, these new item types tend to be based on complex, problem-solving cases involving the Windows 2000® operating system. The MCP examinations also include traditional multiple-choice and extended matching questions.

This paper summarizes some of the technical aspects of item-level scoring and associated item analysis procedures involving the IIT exercises used on the MCP examinations. For the purposes of this paper, *item-level scoring* is defined as the process of codifying the response(s) and other relevant information that the examinee provides into a numerical quantity that reliably and validly represents the examinee's performance. For example, a traditional multiple-choice item codifies everything the examinee does into a simple item score of one if correct or zero if incorrect. The challenge with more complex item types lies in codifying the relevant data and information that the examinee provides into a statistically valid and reliable score.

The challenges of scoring complex, computer-based performance exercises are certainly not new. Many existing computerized performance exercises involve complex responses and/or performance actions that need to be captured and codified, with the most salient information extracted for scoring purposes (Bejar, 1991; Clauser, et al, 1997; Bennett & Bejar, 1999). Luecht & Clauser (1998) evaluated response formats and scoring for ten operational computerized tests covering skills such as writing, computer programming, architectural design, and medical patient management. In many cases, the supposedly "complex" computerized performance tests require responses that are simply discrete entries from a list of possible entries. In other cases, the source inputs really are more complex – perhaps constructed responses, essays and other extended text entries – that need to be processed and scored using fairly complicated algorithms. As we move further along the continuum of possibility in the realm of computerized testing, we will see even more complex performance and response data such as speech recognition requiring neural nets for processing (e.g., Bernstein, DeJong, Pisoni and Townshend, 2000). On a complexity continuum, the IIT exercises lie somewhere between discrete items and some of more variable free-form entries alluded to above.

This paper is divided into two sections. In the first section, I present a simple, but generalizable framework for understanding scoring systems and the underlying data structures for those systems. The scoring system for the MCP examinations is an instantiation of that framework. In the second section, I

present some operational details about how that scoring framework was implemented at Microsoft, including examples to illustrate various aspects of the framework. Finally, in the discussion section, I will layout some basic design principles that may help generalize this framework to other complex computerized exercises.

A Framework for Scoring Complex, CBT Items

There are two basic components in most scoring frameworks: (1) responses provided by the examinee and (2) answer expressions. The actual scoring rubrics used by a particular testing problem may also involve special transformations of the raw response data (e.g., parsing, recoding, or even modeling via a neural net to produce a set of quantitative values) and compound logic rules, however, these two basic components are reasonably broad in scope and can be shown to extend to many different types of test items and performance exercises.

Examinee Responses

Responses are generated by actions or inactions taken by the examinee and can vary across a wide spectrum of complexity (Luecht & Clauser, 1998). At one end of the spectrum are simple text-character responses (e.g., "A"), short answers, or special component-event actions (e.g., Radio-Button1.Value=*True*). Most modern computerized tests provide **controls** that can capture these simple

response actions using a state indicator and/or a result value that the control passes back as the response. For example, to record whether or not an examinee uses a mouse to "click" a particular control object such as a check box displayed on the computer, we could record as the response variable of interest the state of the control object (e.g., given some generic control, we could record *generic_control.state*=**True** or *generic_control.state*=**False**, corresponding to the "on" or "off" state). We could likewise store the result of a keystroke or edited text entry (e.g., *generic_control.value*="My response") At the other end of the spectrum, there are more complex data structures captured as examinee inputs (e.g., formatted essay text, computer programs, mathematical proofs, speech samples) that usually require substantial transformations to simplify the data. From an abstract perspective, we can think of such custom controls as transformation functions of the examinees' response—i.e., *custom_control.value=f*(**response set**).

Inactions or non-response can also constitute a response of sorts. In most cases, we can represent the inaction as a null state or value for the response control.

Some of the customized response coding procedures we will discuss here involve determining the nature of the connection between two objects or "nodes" ("drag-and-connect" items), determining the serial position of various response options relative to other response options (i.e., for "build-list-and-reorder" items). The actual response is stored as a function of the item for a particular

examinee on a particular test. One way of hierarchically expressing the storage structure is as follows:

examinee.test.item.response.state=generic_control.state

and/or

examinee.test.item.response.value=generic_control.value.

Once the *response.state* and/or *response.value* are recorded, either or both can be scored. Items may have a single response or multiple responses. Multiple responses can be stored as multiple, simple transactions between the examinee and the item, or, as complex transactions comprised of compound responses interactions between the examinee and the items.

Answer Expressions

Answer keys and other rubrics are idealized responses or patterns of responses that can be linked, in whole or part, to an examinee's response in the form of a transaction. Often, the transaction resolves to "true" or "false" (i.e., a Boolean comparison operation) and we than assign point values to either or both alternatives. For example, dichotomous scoring of multiple-choice items typically uses the logic of comparing a response value (character or state) to a stored answer for each items. If the response matches the answer, a stored value (usually one) is assigned, otherwise the examinee receives no points (a value of zero).

At a slightly more technical level, there are really three parts to what we can call an **answer expression**: (a) the response or response function being evaluated; (b) the answer key, answer key sets, or rubrics used to compare to the responses that we have decided to score; and (c) a logical grammar (e.g., Boolean) that carries out the comparison and evaluates to a particular value. For this paper, I will limit the comparative grammar discussion to standard Boolean algebra, which covers all of the Microsoft IIT exercises.

One important point to make is there may be many responses, actions or inactions that we choose to ignore[2]. Furthermore, answers are not always "correct". Sometimes answers denote proper responses the examinee should make (e.g., the usual "correct" answer or a beneficial problem-solving action); other times answers denote responses the examinee should not make (i.e., so-called "dangerous" options). The most obvious type of answer key is a multiple-choice answer key – stored in an item database – which may hold be a single character or serial position index referring to one element in a vector of responses. Answers can also refer to compound response objects, actions or inactions, patterns of responses, or combinations of actions and responses. For example,

item.answer.value="A" + "1" + "C"

[2] A multiple-choice answer is a prime example of ignoring certain responses. We score the correct answer and essentially ignore all incorrect answers.

is a compound answer[3]. Obviously, more elaborate storage schemes can be devised, using array structures, etc..

Answer keys may also be expressed as functions of *other* responses. This approach, while not specifically used for the MCP examinations, allows some answers to be dependent on other responses or functions of responses that the examinee provides. For example, a calculated response in one part of a performance exercise could be used in future calculations to measure whether an examinee followed proper procedures and decision-making, regardless of whether or not the original calculated response was correct or incorrect. If we denote the other data used in the answer as "other_parameters.values" (for scoring), then the answer can be expressed as

item.answer.value=f(examinee.item.response.variable, other_parameter.values).

Obviously, from a certain psychometric viewpoint[4], specifically allowing dependencies among response may be undesirable, but the system ought still be capable of dealing with them, regardless.

In the remainder of this paper I will use the more generic notation, *"item.expression"* in referring to "item answers". This avoids the typical connotations of answers as always referring to correct responses.

---

[3] This type of concatenation usually assumes some similar transformation function applied to the raw response data to make it comparable—e.g., item.response.value=*f* (examinee.response).
[4] It is fairly well established that dependencies among scored responses can lead to attenuated reliability in the classical testing tradition and to violations of the local independence assumption required for certain item response theory models.

As noted above, comparing a discrete response to a single answer key using a Boolean "IF *f(x,y)* THEN *assign(value)*…" is the typical method of scoring traditional multiple-choice questions, where *assign(value)* is usually an equality such as *item.score=item.expression*.**value**. Of course, I am assuming that an *item.expression*.**value** of one or some other value or weight has been stored in the item database. For example, suppose that an examinee answers "A" to a test question (item). The scoring system records the response and then matches that input response to the answer key stored in a database. If the answer key is likewise "A", the response-to-answer comparison resolves as "true" and a specified point value is assigned. Assuming dichotomous scoring, a score of u_i = 1 is assigned to the examinee's response if the response matches the answer key, otherwise a [default] null score of u_i = 0 is assigned.

We can write the implicit scoring rule as a generalized boolean statement,

IF *item.response=item.expression-x*
THEN *item.score* = *item.expression*.**value**

where *item.expression-x*.**value** allows for multiple expressions for a given item and corresponding ".**value**" statements to be stored as part of the "*item.expression-x*" database entity (x=1,…,n expressions for a given item). It is interesting to note that the incorrect multiple-choice "distractors" can be assigned individual expression scores as well by simply: (i) assigning additional expressions identifiers for each item distractor; (ii) writing for each expression a Boolean logic statement with an assigned "answer" (in this case, an incorrect

response state or value representing each distractor); and (iii) setting the value of the expression, if it resolves to "true".

An important IIT achievement was designing a flexible, hierarchical database representation of the item expressions. That is, the simple "item" + "expression" construct can be attached to either an examinee test record (e.g., *examinee.test.item.expression*) or stand on its own as a item database entity (*itembank.item.expression*). The "expressions" were also formalized under IIT as Boolean logic statements, stored with the individual expression for each item[5].

For the visually minded, Figure 1 shows a simple database "table" to illustrate storage of the item expressions, assuming partial-credit scoring, for a four-option, multiple-choice item with two "correct" answers, "A" and "D". By extension, if we wanted to only assign a point for getting both answers "A" and "D" and nothing for just "A" or "D", we could replace expressions 99999.001 and 99999.004 with the expression, 99999.response=("A" AND "D"), and assign a score value of one or two.

Insert Figure 1 Here

Of course, expressions "99999.002" and "99999.003" (the distractor scoring rules with weights of zero) add nothing to the test scores. We could develop a

[5] In principle, this mechanism can be generalized to any grammar that can be "scripted" or even delegated to customized "scoring agents", called by name and passed response and answer-based "messages" (e.g., a neural net). Each script or agent must simply return a value (logical or real-valued). This capability, of course, will probably be active in Version 2.0 of IIT.

bit more elegant logic by simply replacing the distractor scoring rules with a negation of the compound rule:

IF **NOT**(*item.response*="A" OR *item.response*="D")
THEN *item.score* = *item.expression*.**value.**

Implementation of Scoring for IIT

The scoring engine used by Microsoft is proprietary and secure, for obvious reasons. However, the principles implemented by that scoring engine are a direct instantiation of the scoring framework described in the previous section.

The database structure for the items and answer expressions (see Figure 1, above) was an important development for IIT. However, there were actually three processes that allowed Microsoft to implement scoring for IIT. First, there needed to be a process for identifying the critical response objects or scorable functions of those response objects (e.g., selecting the proper "nodes" and relationships among those "nodes"). Second, there needed to be a set of procedures for efficiently creating the item answer expressions to represent all potentially scorable incidents. Third, there had to be a way to evaluate and choose the more effective item answer expressions from multiple possibilities.

Identifying the Critical Response Functions

In the case of a multiple-choice item, identifying the "response objects" is a rather simple process. Ask the item writer to identify the correct answer. However, for many of the Microsoft IIT exercises, the response options tend to be multi-faceted, involving relations among various objects or nodes.

Table 1 contains a breakdown of the primary response components for the current crop of IIT exercises. This table lists the five primary IIT exercise types and the response objects for each: (1) one-best answer multiple-choice items (MC^1), (2) extended matching and multiple-best answer multiple-choice items (MC^k); (3) create-a-tree (CT); (4) build-list-and-reorder (BLR); and (5) drag-and-connect (DC). The MC^1 and MC^k items are very similar to standard multiple choice items where the examinee chooses one or multiple responses[6].

Insert Table 1 Here

[6] One note is important regarding the number of response choices that an examinee is allowed for multiple-pick items (MC^k). Although it is tempting to allow the examinee to choose "all that apply", such an unconstrained multiple-choice response format has three consequences. First, it exponentially increases the viable number of combinations of scorable responses. This is easy to show by an example. Given five answer choices and the mandate to choose two, there are exactly five-take-two (i.e., $C_2^5 = 10$ response combinations to consider) answer combinations to consider. Allowing the examinee to choose "all that apply" requires evaluating $C_0^5 + C_1^5 + C_2^5 + C_3^5 + C_4^5 + C_5^5 = 32$ combinations. There are obvious storage and processing implications. Second, the "choose-all-that-apply" response format makes identifying the most critical scorable incidents per item intractable for item writers or scoring committees. Imagine having item writers evaluate the full combination of responses for an item with 12 options – there happen to be 4,096 viable combinations. Finally, "choose-all-that-apply" items require a scoring solution to check that examinees simply don't choose all the answers (that is, m take m, possibly checking for m take $m-1$ solutions, as well). In any case, this seemingly simple capability of allowing the examinee to choose "all that apply" usually necessitates messy solutions to a problem that can be avoided by simply having the examinees choose a particular number of options and having the computerized response handler constrain the number of selections to particular number ($k < m$).

The CT items require the examinee to choose from a list of possible choices and align those choices under a tree-like set of nodes (Fitzgerald, 2001). The scoring model is extensible to allow multiple layers of the tree, similar to an outline (e.g., 1.1.1, 1.1.2, etc.). When limited to a two layers (e.g., 1.1, 1.2, 2.1, 2.2,...), CT items are essentially extended matching items. The BLR items are primarily intended to measure an examinee's ability to select and prioritize a set of options. The relationship between any pair of objects can be defined as an ordered list[7] (e.g., "AB" is correct, "BC" is not). It is possible to represent "triples" and higher-order combinations in the final response list by identifying the objects and their serial position or their position relative to one another.

The DC items consist of two objects, joined pair-wise, be a particular relationship. The objects can be network components or any relevant objects. The relationships can be classified as directional or bi-directional and are typically represented as line or arc "connectors" between the objects (Fitzgerald, 2001). Other relations can include types of real connectors (e.g., communication or transmission protocols between networks). Higher order joins among the objects are also possible (e.g., triple clusters comprised of three objects and up to three connectors). By identifying the objects and connections (relations) in separate lists, standard graph theory can be applied to identify the "nodes" and "edges" that form a *scorable incident*.

---

[7] The actual scoring mechanism used by Microsoft is a bit more sophisticated and efficient than using ordered lists, however, the result is about the same.

Creating the Item Expressions

Given the enormous potential for large numbers of answer expressions for the MC^k, CT, BLR, and DC items, a process had to be developed to efficiently produce, encode and try out viable expressions by scoring real examinee data. The solution was actually quite simple and has been effectively used with a number of MCP examinations over the past year.

Initially, the item writers generate lists of viable expressions to be scored. Then, a committee of subject-matter experts (SMEs)—including the item authors, if possible—is convened and given two simple tasks. First, they are instructed identify the response choices or actions that an examinee MUST do—i.e., "good" things. These are deemed *positive* actions and are assigned "+" on a coding sheet[8]. The SMEs are trained to use Boolean statements (or other custom operator functions understood by the Microsoft scoring engine) to express positive response choices or actions. Second, the SMEs identify the response choices or actions that an examinee MUST NOT select or perform. These negative actions are critically "bad" things and are assigned "–" on the coding sheet, once the expression is created. Item expressions in the initial lists that are not chosen as "good" or "bad" are dropped. The SMEs are then given the opportunity to add additional plus or minus expressions to the list. A final screening is done to attempt to reduce the list of expressions to a fixed number of

n or fewer expressions, where *n* is fixed as a "strongly recommended policy" for the upper bound.

Examples of MC[1] and MC[k] item expressions have already been provided. An example of two CT item expressions might be: 88888.001: "B2" and 88888.002: "B6" which identifies nodes **2** and **6** as child nodes of a parent node, **B**. Other nodes can be similarly expressed as relate to other "parents" (e.g., A or C parent node objects). Unions of such relations can express the need to have both assigned to the same parent; for example, 88888.003: "B2 AND B6". Other graphs of the relation are possible (e.g., expanding the implicit binary tree to include "cousins" 2 and 6 and coding the expressions accordingly).

An example of a BLR item expression is 77777.001: "B" BEFORE "A", where the "BEFORE" operator evaluates the relative sequencing of selected objects in the response list and resolves to a Boolean result. There is no requirement that the response objects be adjacent in the list, although that level of specificity can certainly be enforced by using compound statements. Other mechanisms for evaluating absolute and relative serial position of items in a list are, of course, possible. Finally, we come to the drag-and-connect item expressions. An example of a DC item expression for a pair connected pair of objects would be 66666.001: "B & 2 & D", which shows response objects "B" and "D" connected

---

[8] Initial work with the SME-committees indicated that they could not efficiently come to consensus about logically derived scoring weights for the expressions and spent enormous amounts of time in discussion, nonetheless.

(joined) by relation "2". Other higher-order object connections could be represented by multiple pair-wise connections, triples, etc..

The weight values for the item expressions were assigned using a scoring protocol termed *Dichotomous Expression Weighted Scoring* (DEWS; Luecht, 2000). Under the DEWS protocol, examinees only acquire positive points for doing positive things OR for not doing negative things. In addition, many feasible incidents can be treated as "neutral" and not scored at all (i.e., implicitly assigned a weight of zero).

Specifically, positive scorable incidents (expressions) are assigned weights of one. Negative incidents are recoded as "not doing a bad thing" and likewise assigned weights of one. Recall that negative incidents involve choosing incorrect answers or selecting actions that are *critically* wrong. By inverting the logic of negative incidents and assigning NOT operator to them, it is feasible to give credit to everyone for not doing something bad, rather than penalizing the examinees for the negative action or answer choice. The logical inversion effectively eliminates negative weighting. The basic Boolean logic is

IF NOT(*item.response=item.expression-x*)
THEN *item.score* = *item.expression*.**value**

where *item.expression*.**value** = 1 for both positive and the complementary NOT() of negative expressions. Therefore, examinees get points for doing good things and for not doing bad things.

There are two practical advantages of the DEWS protocol. First, DEWS maps directly into most partial-credit scoring models using number-correct scoring or item response theory, where the item expression scores are additive over expressions, summing to a polytomous score for each item. Second, where necessary, it is possible to reconfigure the expression lists for a given item and rescore the examination response data, without major complications. Luecht (2000) discussed several other technical advantages of DEWS over using other weighting schemes, including some reasons to avoid negative weights typically associated with "dangerous options".

Evaluating the Effectiveness of Item Expressions

The item expression scores (weight values) are additive in the items and, by extension, in the test scores. As a result, it is possible to use **standard item analysis** procedures to evaluate the contribution of individual item expressions (e.g., expression percent-correct scores, high-low group percentages, and expression-test correlations).

In keeping with usual practices, item expressions that are uninformative or behaving poorly are typically discarded. This serves two purposes: (1) it streamlines the number of expressions, leaving only the most statistically useful expressions; and (2) over time, the process of identifying and coding expressions can ideally be improved by considering what works with the current items.

Figure 2 shows a sample item-expression analysis report for a MC[k] item. The top section of the report aggregates the expression-level statistics and reports the item mean, standard deviation and item-test correlation. Note that the item-test correlation is a product moment correlation between the expected response function and the total score, where the latter has been properly adjusted to remove any autocorrelation with the current item score[9]. The graphic displays the mean item scores for "quintiles" (0 to 20th percentile, 21st to 40th percentile, etc.). The lower section of the report displays the expression-level summary. The values in the columns labeled "Q1", "Q2", etc., correspond to the proportions of examinees receiving credit for each expression within each of the quintiles. Most of the expressions seem to be performing well insofar as showing a monotonic increase in the expression scores as we move from Q1 to Q5. The expression for answer "C" (see boxed section) is somewhat less informative that might be desired, since the scores actually decrease slightly at Q3 and Q4. By reviewing information like this, informed decisions can be made about which expressions to retain and which should be discarded.

Insert Figure 2 Here

Figure 3 shows a sample item-expression analysis report for a build-list-and-reorder (BL) item. The complexity of the expression logic (see callout in the

[9] A more recent version of the IIT item analysis software also reports expression-test product-moment correlations to more quickly detect near-zero or negatively discriminating expressions.

lower section of Figure 3) includes both relative sequencing logic and some negative options that the examinee should not include in the list ("C" and "D"). Considering the proportions across the quintiles, the first [boxed] expression, "xxxxxx.01" is less informative than the second expression. Nonetheless, both expressions are contributing positively to the total score.

Insert Figure 3 Here

Figure 4 shows a sample drag-and-connect (DC) item-expression analysis report. The second expression, "xxxxxx.02", illustrates how the scoring engine will give credit for not choosing a particularly complex negative set of nodes and relations. Both expressions seem to work reasonably well, although the item is somewhat difficult (i.e., the item percent-correct is 0.703 divided by a maximum of 2 points, which is equivalent to a *p-value* of 0.35).

Insert Figure 4 Here

Discussion

The IIT scoring process established for the MCP examinations provides some important examples of procedural and system-level components. The process may be specific to the MCP item types, however, there general principles that that seem necessary when developing an integrated system for scoring computer-based performance items.

First, the formal design of the supporting database structures for the scoring system needs to be done early on in the process. This cannot wait until the test is nearly operational and the data are rolling in. For the IIT system, the table-based structure (see Figure 1) proved to be a simple, yet elegant way of representing the answer expressions and associated scoring weights. That same structure also seems to generalize well for almost any items that use *transactional* scoring protocols. It can further be shown to extend to more complex scoring models using functions of the responses and answer values or other data stored in an item-level database.

Second, the scoring system needs to *flexibly* represent complex response relationships and compound logic denoting performance expectations. The discussion of answer expressions and some of the specific examples shown rather clearly demonstrate that inherent capability of the IIT system.

Third, the process of developing answer keys and answer expressions needs to be efficient, especially if new items are produced *en masse* to support on-going computerized testing. This is one area that can be incredibly costly in terms of human capital (item writers, SME committees, test editors, etc.). By moving to a simple decision model for identifying critical positive and negative actions and responses, combined with empirical summary data to evaluate the individual expressions (i.e., the item-expression analyses), the MCP/IIT process has become highly efficient. It currently takes only a day or two to complete for an entire item pool.

Finally, the scoring process needs to consider additional scaling or calibration needs. In the present context, the expressions sum at the item level to produce polytomous scores. Partial-credit IRT calibrations can naturally proceed, using those scores. Because of the pre-screening of the expressions (i.e., using the item-expression analyses to sanitize the final set of expressions per item), the final scores will tend to be well-behaved. This becomes an additional bonus when attempting IRT calibrations. Although not reported in this paper, a number of calibrations performed by the author on the "cleaned" item-expression data have consistently shown exceptional fit of all of the items, using the Rasch-based partial credit model.

References

Bejar, I. I. (1990). A methodology for scoring open-ended architectural design problems. *Journal of Applied Psychology, 76*, 522-532.

Bennett, R. E. & Bejar, I. I. (1999). Validity and automated scoring: It's not only the scoring. *Educational Measurement: Issues and Practices, 17*, 9-17.

Bernstein, J. , DeJong, J., Pisoni, D., & Townshend, B. (2000). Two experiments on automatic scoring of spoken language proficiency. In P. Deleloque (Ed.) *Proceedings of Integrated Speech Technology in Learning 2000.* Dundee, Scotland: University of Abertay, August 2000, pp. 57-61.

Clauser, B. E., Margolis, M. J., Clyman, S. G., & Ross, L. P. (1997). Development of automated scoring algorithms for complex performance assessments: A comparison of two approaches. *Journal of Educational Measurement, 34*, 141-161.

Fitzgerald, C. (2001, April). *Rewards and challenges of implementing an innovative CBT certification examination program.* Paper presented at the Annual Meeting of the National Council on Measurement in Education, Seattle, WA.

Luecht, R. M. (2000, April). *Microsoft® Windows® 2000 Design Examinations A Study of Scoring Protocols.* Unpublished Microsoft Whitepaper.

Luecht, R. M. & Clauser, B. E. (1998, September). *Test models for complex computer-based tests.* Paper presented at the ETS colloquium Computer-based Testing: Building the Foundation of Future Assessments, Philadelphia, PA.

Contact Information

Professor Richard M. Luecht
University of North Carolina at Greensboro
Dept. of Educational Research Methodology
Curry Building 209
P.O. Box 26171
Greensboro, NC 27402-6171

Telephone: 336.334.3473
Email: rmluecht@uncg.edu

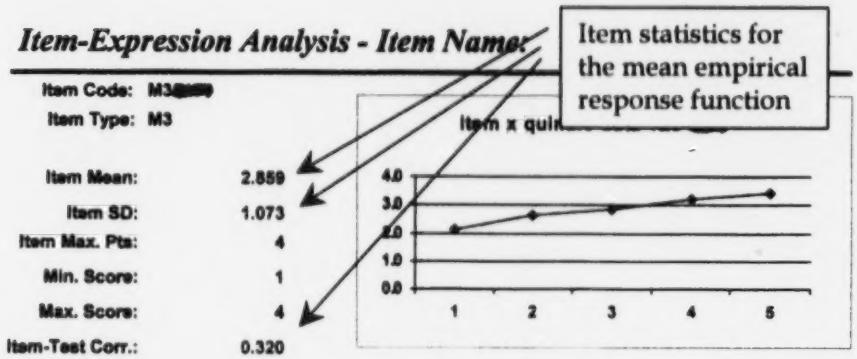
Table 1. Scorable Response Components for IIT Items

Item Type	Description of Response Options
MC[1]	Choose one of *m* response control (*M*=max. distractors)
MC[k]	Choose *k* of *m* response control (*m*=max. distractors, $k \leq m$)
CT	Move *k* of *m* "node" objects to slots below *r* of *p* parents to form a hierarchical tree (i.e., *parent* and *child* relations) ($k \leq m$; $r \leq p$)
BLR	Select and serialize *k* of *m* objects in an ordered list ($k \leq m$)
DC	Connect *k* of *m* objects ($k \leq m$) via *c* of *d* relations ($k \leq m$; $c \leq d$)

Notes: [1] One-best answer multiple-choice items.
[k] Multiple-best answer, multiple-choice items.

Item ID	Expression ID	Expression	Value
99999	001	99999.response = "A"	1
99999	002	99999.response = "B"	0
99999	003	99999.response = "C"	0
99999	004	99999.response = "D"	1

Figure 1. A Sample Database Table for a Four-Option MC Item with Two Correct Answers



Item-Expression Analysis - Item Name:

Item Code: M3[redacted]
Item Type: M3

Item Mean:	2.859
Item SD:	1.073
Item Max. Pts:	4
Min. Score:	1
Max. Score:	4
Item-Test Corr.:	0.320

Item x qui[redacted]

4.0
3.0
2.0
1.0
0.0
1 2 3 4 5

Item-Expression Code	Expression-Level Statistics							
	Mean	SD	N	Q1	Q2	Q3	Q4	Q5
M3[redacted]01 *Expression:* A	0.844	0.363	256	0.65	0.84	0.90	0.90	0.92
M3[redacted]02 *Expression:* B	0.512	0.500	256	0.14	0.29	0.61	0.71	0.81
M3[redacted]03 *Expression:* C	0.813	0.390	256	0.84	0.80	0.75	0.78	0.88
M3[redacted]04 *Expression:* NOT(D OR E)	0.691	0.462	256	0.51	0.69	0.63	0.80	0.83

Figure 2. A Sample Item and Expression Analysis Report for a MC[k] Item

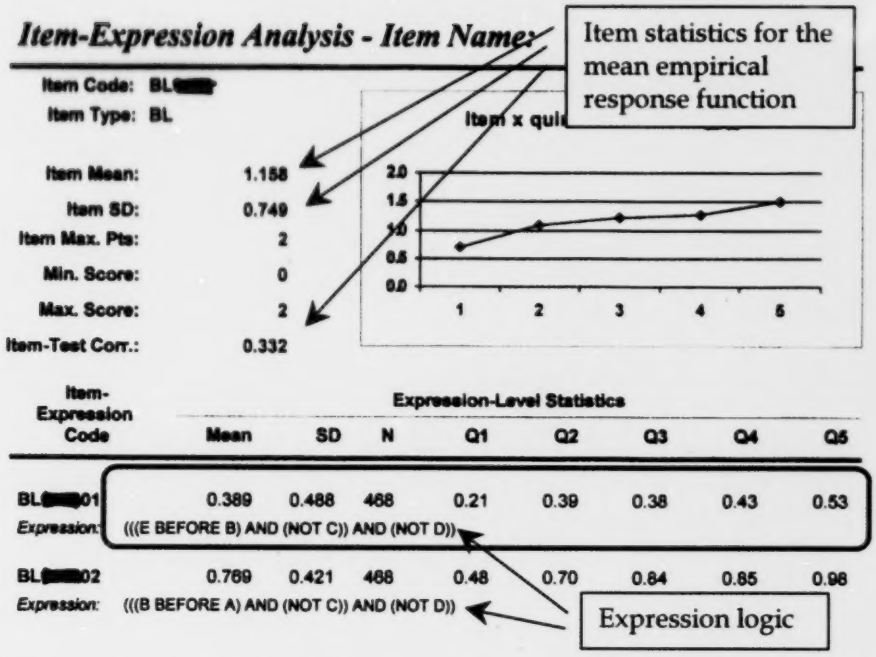


Figure 3. A Sample Item and Expression Analysis Report for a BL Item

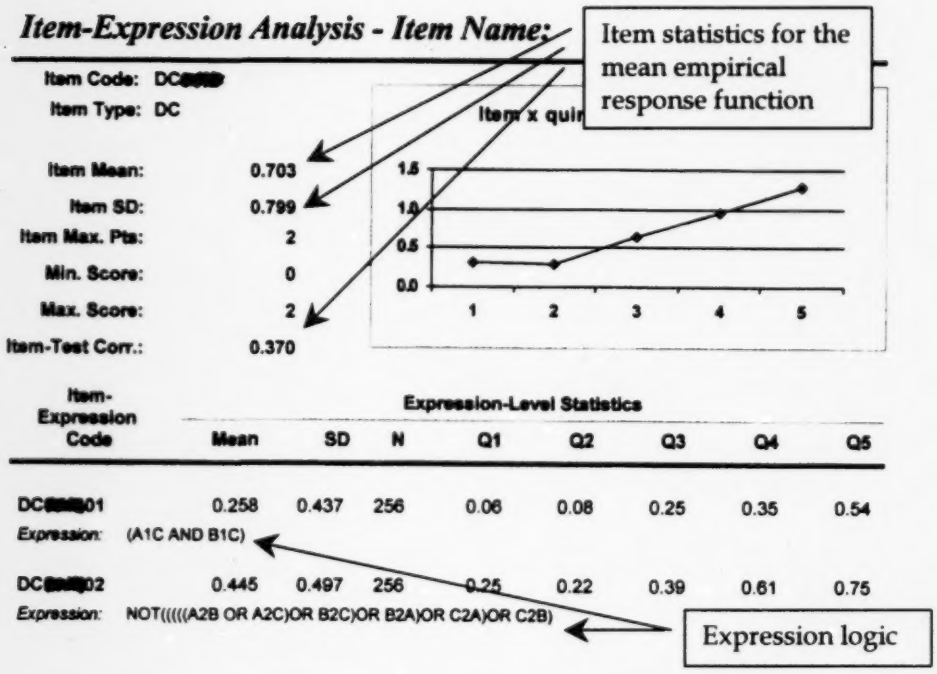

Item-Expression Analysis - Item Name:
Item statistics for the mean empirical response function
Item Code: DC
Item Type: DC
Item Mean: 0.703
Item SD: 0.799
Item Max. Pts: 2
Min. Score: 0
Max. Score: 2
Item-Test Corr.: 0.370
Item-Expression Code
Expression-Level Statistics
Mean SD N Q1 Q2 Q3 Q4 Q5
DC 01 0.258 0.437 256 0.06 0.08 0.25 0.35 0.54
Expression: (A1C AND B1C)
DC 02 0.445 0.497 256 0.25 0.22 0.39 0.61 0.75
Expression: NOT(((((A2B OR A2C)OR B2C)OR B2A)OR C2A)OR C2B)
Expression logic

Figure 4. A Sample Item and Expression Analysis Report for a DL Item